## Project Overview: Sign Language Recognition using Spatiotemporal CNN + Mediapipe

**Scope**

This System will be Beneficial for Both Dumb/Deaf People and the People who do not understand sign Language. They just need to do that with sign Language gestures and this system will identify what he/she is trying to say after identification it gives the output in the form of the predicted class.

### Goal:

To classify videos into **4 sign classes**:  
 ['Afternoon', 'Night', 'Morning', 'Evening']  
 by learning from **temporal patterns of body and hand landmarks**.

**Project Requirements :**

1. **Hardware Requirement:**

* **Webcam**

1. **Software Requirement:**

* **Operating System:** Windows 8 and Above

1. **IDE**: Jupyter Notebook
2. **Programming Language:** Python 3.9 5
3. **Python libraries:** OpenCV,NumPy ,Keras ,mediapipe ,TensorFlow

**Setup Instructions**

1. Clone the Repository.
   * git clone **https://github.com/tanyanebhwani/isl-organised**
   * cd isl-organised
2. Install Dependencies.
   * Use pip to install all required libraries:
   * pip install numpy scikit-learn seaborn matplotlib opencv-python mediapipe tensorflow
3. Now If you are on **Jupyter Notebook,** run the following command**:  
    %run ./userInterface.py**
4. Use a spacebar to record the sign and you will get predicted class on the top right corner.

**Folder Structure**

**isl-organised/**

├── **main.ipynb** # Main notebook for preprocessing, training, and evaluation

├── **userInterface.py**  # Real-time webcam prediction script

│

├── **feature\_extraction/**

│ └── **feature\_extraction.py** # Extracts frames and prepares training input X.npy

│

├── **mediapipe/**

│ └── **mp.py** # Converts original videos to Media Pipe skeleton videos

│

├── **models/**

│ └── **model.py** # Defines and compiles the Spatiotemporal CNN model

│

├── **data/**

│ ├── **Days\_and\_Time/** # Raw input videos, sorted by class (Morning, Night, etc.)

│ └── **Days\_and\_Time\_skeleton/** # Skeleton videos created using Media Pipe

│

├── **X.npy** # Input features: (num\_videos, 30, height, width, 3)

├── **y.npy** # Labels: (num\_videos,)

├── **my\_sign\_model.h5** # Trained model weights

│

└── **README.md** # (Optional) Project documentation

## How the Project Works (Step-by-Step)

### 1. Preprocessing (in main.ipynb)

#### a. MediaPipe Landmark Extraction

* + - Each raw video from **data/Days\_and\_Time/** is passed to **use\_mediapipe()** (from **mediapipe/mp.py**)
    - This function creates a "landmark video" using MediaPipe — a visual representation of key skeleton joints.
    - Output videos are saved to: **data/Days\_and\_Time\_skeleton/**

Format: **MVI\_<video\_id>\_skeleton.mp4**

#### b. Frame Extraction and Feature Preparation

* The function **combine\_features()** (from **feature\_extraction/feature\_extraction.py**) is called on the landmark videos.
* Extracts 30 frames per video.
* Converts them to RGB, resizes if needed
* Combines them into a NumPy array X with shape:  
   **(num\_videos, 30, height, width, 3)**
* Saves this as **X.npy.**

#### c. Label Assignment

* + The labels y are assigned corresponding to the folder/class the original video belongs to (Morning, Afternoon, Evening, Night)
  + Saved as **y.npy** with shape: **(num\_videos,)**

**2. Train-Validation-Test Split**

* First split:
* **X\_train\_val, X\_test** → 80% for training + validation, 20% for testing
* Second split (from **X\_train**):
* **X\_train, X\_val** → 90% train, 10% validation

**3. Model Training (Spatiotemporal CNN)**

* Model defined in **models/model.py**:
* Input shape: **(30, height, width, 3)**
* Learns both spatial (hand shape) and temporal (movement over time) patterns.
* Trained on:
* **X\_train, y\_train**
* Validated on: **X\_val, y\_val**
* Metrics used:
* Accuracy, confusion matrix, classification report
* Trained model saved to:  
   **my\_sign\_model.h5**

**4. Real-Time Inference (in userInterface.py)**

#### a. Frame Collection

* Uses webcam to capture live frames
* For each frame:  
  + Media Pipe is applied to detect skeleton.
  + Landmarks are drawn on a white canvas (creating a frame like those used in training).
  + This canvas is stored in a **deque buffer**

#### b. Prediction

* + - Once the buffer has 30 frames:
    - Frames are converted into shape:  
        
       **(1, 30, height, width, 3)**
    - Passed to the trained model
    - Predicted class is displayed on the screen (e.g., "Morning", "Night", etc.)

**Contact**

For any questions or suggestions:

**1. Email**: [nebhwanitanya0@gmail.com](mailto:nebhwanitanya0@gmail.com)

**2. GitHub**: [tanyanebhwani](https://github.com/tanyanebhwani/)